computation sometimes requiring dynamic scheduling. The *switch* and *select* actors allow conditional dataflow statements with semantics for control flow as shown in Figure 2.

3. Please replace the paragraph starting on page 2, line 13, with the following paragraph. 5/23/08 T.T.

Cyclo-dynamic dataflow, or CDDF, as defined by Wauters et al. (see P. Wauters, M. Engels, R. Lauwereins, J.A. Peperstraete, "Cyclo-Dynamic Dataflow," p. 0319, 4th Euromicro Workshop on Parallel and Distributed Processing (PDP '96), 1996), extends cyclo-static dataflow to allow run-time, data-dependent decisions. Similar to a CSDF, each CDDF actor may execute as a sequence of phases *f1, f2, f3, f4, ..., fn*. During each phase *fn*, a specific code segment may be executed. In other words, the number of tokens produced and consumed by an actor can vary between firings as long as the variations can be expressed in a periodic pattern. CDDF allows run-time decisions for determining the code segment corresponding to a given phase, the token transfer for a given code segment, and the length of the phase firing sequence. The same arguments for using caller prediction for BDF graphs can be extended to include CDDF.

4. Please replace the paragraph starting on page 2, line 28, with the following paragraph. 5/23/08 T.T.

Lee and Parks (see E. A. Lee and T. M. Parks, "Dataflow Process Networks," Proceedings of the IEEE, pp. 773-801, 1995) describe dataflow process networks as a special case of Kahn process networks. The dataflow process networks implement multiple concurrent processes by using unidirectional FIFO channels for inter-process communication, with non-blocking writes to each channel, and blocking reads from each channel. In the Kahn and MacQueen representation (see G. Kahn and D. B. MacQueen, "Coroutines and Networks of Parallel Processes," Information Processing 77, Gilchrist

ed., North-Holland Publishing Co., 1977), run-time configuration of the network is allowed. While some scheduling can be done at compile time, for some applications most actor firings must be scheduled dynamically at run-time.

5. Please replace the paragraph starting on page 3, line 20, with the following paragraph.

Figure 4 illustrates two-level branch prediction. A '1' in the counter MSB predicts that a branch will be taken, while a '0' predicts that the branch will not be taken. This approach may achieve a prediction success rate in the 90% range. Patt and Yeh (*see* T.-Y. Yeh and Y. N. Patt, "Two-level adaptive training branch prediction," in Proceedings of the 24th Annual ACM/IEEE International Symposium on Microarchitecture, pp. 51-61, 1991) have tabulated the hardware costs to be significant for large history lengths.

6. Please replace the paragraph starting on page 10, line ~~20~~ 6, with the following paragraph. 5/23/08 T.T.

There are a few characteristics of applications that can take advantage of caller prediction model for prefetching node instance data. In one embodiment, applications should not execute in hard real-time since the prediction logic may be wrong, which may result in jitter. In one embodiment, applications for run-time caller prediction may include applications where quality of service (QoS) can vary depending on how fast the application is running. Buttazzo et al. (*see* G. Buttazzo and L. Abeni, "Adaptive Rate Control through Elastic Scheduling," in Proceedings of the 39th IEEE Conference on Decision and Control, pp. 4883-4888, 2000) point out that these applications are common due to the non-deterministic behavior of common low-level processor architecture components, such as caching, prefetching, and direct memory access (DMA) transfers. Buttazzo's work suggests voice sampling, image acquisition, sound generation, data compression, video playback, and certain feedback control systems as application

domains that can function at varying QoS depending on the execution rate of the application.

7. Please replace the paragraph starting on page ~~3~~ 10, line ~~20~~ 17, with the following paragraph. 5/23/08 T.T.

Feedback control systems are particularly interesting because they may contain several subsystems that share common components. Abdelzaher et al. (*see* T. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control," in IEEE Transactions on Computers, pp. 1170-1183, 2000) explore a complex real-time automated flight control system that negotiates QoS depending on the load of the system. The flight control system consists of a main function that controls the altitude, speed, and bearing of the plane, and also contains subsystems for flight control performance. Continuous analysis of all of the inputs to achieve optimal control is a large computation task. However, the task of flying a plane can be accomplished with relatively little computation power by carefully reserving resources and by tolerating less than optimal control functionality. PID (Proportional-Integral-Derivative) control is a possible compelling application domain because PID tuning parameters are relatively insensitive to rate changes. Many measurement and control applications may display similar properties of varying acceptable QoS.